第5章 数据库基础

本章学习目标

- 掌握数据库的概念
- 掌握关系型数据库建模
- 掌握 SQL 语句
- 了解 Redis 数据库

随着网络逐渐融入人们的生活，Web 数据库也逐渐显示出它的重要性，数据库在网站的建设中已经成为必不可少的重要部分。例如，客户资料、产品资料、交易记录、访问流量、财务分析等都离不开数据库系统的支持，数据库技术已经成为网络的核心技术，因此，本章主要讲解数据库的相关知识。

5.1 数据库基础

5.1.1 数据库基本概念

数据库(Database，DB)是建立在计算机存储设备上，按照数据结构来组织、存储和管理数据的仓库，可将其视为电子化的文件柜——存储电子文件的处所。用户可以对文件中的数据进行增加、删除、修改、查找等操作，此处所说的数据不仅包含数字，还包含文字、视频、声音等。数据库的主要特点如下。

1. 实现数据共享

数据共享包括所有用户同时存取数据库中的数据，也包括用户使用各种方式通过接口使用数据库，并提供数据共享。

2. 减少数据的冗余度

同文件系统相比，数据库由于实现了数据共享，从而避免了用户单独建立应用文件，减少了数据冗余，维护了数据的一致性。

3. 数据的独立性

数据的独立性包括逻辑独立性(数据库的逻辑结构和应用程序相互独立)和物理独立性(数据物理结构的变化不影响数据的逻辑结构)。

4. 数据实现集中控制

文件管理方式中，数据处于一种分散的状态，不同的用户或同一用户在不同的处理中其文件之间毫无关系。利用数据库可对数据进行集中控制和管理，并通过数据模型表示各种数据的组织以及数据间的联系。

5. 数据一致性和可维护性

主要体现在安全性控制(以防止数据丢失、错误更新和越权使用)、完整性控制(保证数据的正确性、有效性和相容性)和并发控制(在同一时间周期内,允许对数据实现多路存取,又能防止用户之间的不正常交互)上。

6. 故障恢复

由数据库管理系统提供一套方法可及时发现故障并修复,从而防止数据被破坏。数据库系统能尽快解决运行时出现的故障,这些故障可能是物理上或是逻辑上的错误。

另外,数据库不是数据库系统,数据库系统的范围比数据库大很多。数据库系统是由硬件和软件组成,其中硬件主要用于存储数据库中的数据,软件主要包括操作系统以及应用程序等,数据库系统如图 5.1 所示。

在图 5.1 中,展示了数据库系统几个重要部分的关系,具体如下:

数据库管理系统(DataBase Management System,DBMS):指一种操作和管理数据库的大型软件,用于建立、使用和维护数据库,对数据库进行统一管理和控制,以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中的数据。

数据库应用程序(DataBase Application System,DBAS):用户在对数据库进行复杂管理时,DBMS 可能无法满足用户需求,这时就需要使用数据库应用程序访问和管理 DBMS 中存储的数据。

数据库(DataBase,DB):指长期保存在计算机的存储设备上,按照一定规则组织起来,可以被各种用户或应用共享的数据集合。

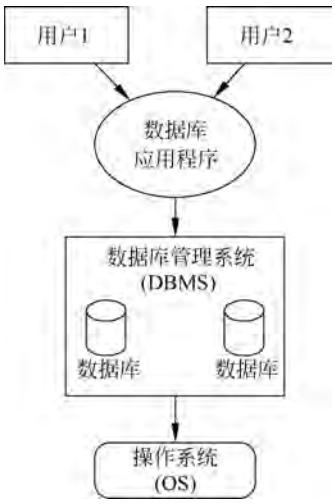

图 5.1 数据库系统

通常情况下,用数据库来表示所使用的数据库软件会引起混淆,确切地说,数据库软件应该是数据库管理系统,数据库是通过数据库管理系统创建和操作的。

5.1.2 常用数据库简介

随着数据库技术的不断发展,数据库产品越来越多。2017 年 8 月,DB-Engines 发布了最新的数据库排行,如图 5.2 所示。

在图 5.2 中,Oracle 居首位,其他数据库也不同程度地受关注,接下来介绍一些常见的数据库产品,具体如下所示。

1. Oracle 数据库

Oracle Database(又名 Oracle RDBMS,或简称 Oracle)是甲骨文公司的一款关系数据库管理系统。它在数据库领域一直处于领先地位,是目前世界上最流行的关系数据库管理系统之一,其系统可移植性好、使用方便、功能强,适用于各类大、中、小、微机环境。它是一种高效率、可靠性好的、适应高吞吐量的数据库。

2. MySQL 数据库

MySQL 是一种开放源代码的关系型数据库管理系统(RDBMS),使用最常用的数据库

330 systems in ranking, August 2017

Rank Aug 2017	Rank Jul 2017	Rank Aug 2016	DBMS	Database Model	Score Aug 2017	Score Jul 2017	Score Aug 2016
1.	1.	1.	Oracle	Relational DBMS	1367.88	-7.00	-59.85
2.	2.	2.	MySQL	Relational DBMS	1340.30	-8.81	-16.73
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1225.47	-0.52	+20.43
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.76	+0.32	+54.51
5.	5.	↓ 4.	MongoDB	Document store	330.50	-2.27	+12.01
6.	6.	6.	DB2	Relational DBMS	197.47	+6.22	+11.58
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	127.03	+0.90	+2.98
8.	8.	↓ 7.	Cassandra	Wide column store	126.72	+2.60	-3.52
9.	9.	↑ 10.	Redis	Key-value store	121.90	+0.38	+14.57
10.	10.	↑ 11.	Elasticsearch	Search engine	117.65	+1.67	+25.16
11.	11.	↓ 9.	SQLite	Relational DBMS	110.85	-3.02	+0.99
12.	12.	12.	Teradata	Relational DBMS	79.23	+0.86	+5.59
13.	↑ 14.	↑ 14.	Solr	Search engine	66.96	+0.93	+1.10
14.	↓ 13.	↓ 13.	SAP Adaptive Server	Relational DBMS	66.92	+0.00	-4.13
15.	15.	15.	HBase	Wide column store	63.52	+0.10	+8.01
16.	16.	↑ 17.	Splunk	Search engine	61.46	+1.17	+12.56
17.	17.	↓ 16.	FileMaker	Relational DBMS	59.65	+1.00	+4.64
18.	18.	↑ 20.	MariaDB	Relational DBMS	54.70	+0.33	+17.82
19.	19.	19.	SAP HANA	Relational DBMS	47.97	+0.03	+5.24
20.	20.	↓ 18.	Hive	Relational DBMS	47.30	+1.10	-0.51

图 5.2　数据库排行

管理语言——结构化查询语言(SQL)进行数据库管理。MySQL 是开放源代码的,因此任何人都可以在 General Public License 的许可下下载并根据个性化的需要对其进行修改。MySQL 因为其速度、可靠性和适应性而备受关注,大多数人都认为在不需要事务化处理的情况下,MySQL 是管理数据最好的选择。

3. SQL Server 数据库

SQL Server 是美国 Microsoft 公司推出的一种关系型数据库管理系统,它是一个可扩展的、高性能的、为分布式客户机/服务器计算所设计的数据库管理系统,实现了与 Windows NT 的有机结合,提供了基于事务的企业级信息管理系统方案。

4. MongoDB 数据库

MongoDB 是一个介于关系数据库和非关系数据库之间的产品,是非关系数据库当中功能最丰富、最像关系数据库的数据库。它支持的数据结构非常松散,是类似 json 的 bjson 格式,因此可以存储比较复杂的数据类型。MongoDB 最大的特点是支持的查询语言非常强大,其语法有点类似于面向对象的查询语言,几乎可以实现类似关系数据库单表查询的绝大部分功能,而且还支持对数据建立索引。

5. DB2 数据库

DB2 是 IBM 公司开发的关系数据库管理系统,有多种不同的版本,如 DB2 工作组版(DB2 Workgroup Edition)、DB2 企业版(DB2 Enterprise Edition)、DB2 个人版(DB2 Personal Edition)和 DB2 企业扩展版(DB2 Enterprise-Exended Edition)等,这些产品基本的数据管理功能是一样的,区别在于是否支持远程客户能力和分布式处理能力。

6. Redis 数据库

Redis 是一个高性能的 key-value 数据库,在部分场合可以对关系数据库起到很好的补充作用。它提供了 Java、C/C++、C#、PHP、JavaScript、Perl、Object-C、Python、Ruby 等客户端,使用很方便。

数据库通常分为层次型数据库、网络型数据库和关系型数据库三种,而在 Web 开发过

程中常用的数据库为关系型数据库，因此本章第 5.2 节将对关系型数据库进行详细讲解。

5.2 关系数据库

5.2.1 关系数据库简介

关系型数据库是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

现实世界中各种实体以及实体之间的联系均用关系模型来表示，关系模型是指二维表格模型，因而一个关系型数据库是由二维表及其之间的联系组成的一个数据组织。当前主流的关系型数据库有 Oracle、MySQL、DB2、Microsoft SQL Server、PostgreSQL、Microsoft Access 等。

5.2.2 关系数据库建模

数据库建模指的是在设计数据库时，对现实世界进行分析、抽象，并从中找出内在联系，进而确定数据库结构的过程。

数据库的建模过程主要有三个阶段：

(1) 概念模型：在了解用户的需求、用户的业务领域及工作情况以后，经过分析和总结，提炼出用以描述用户业务需求的一些概念性内容。

(2) 逻辑模型：将概念模型具体化，实现概念模型所描述的内容，分析具体的功能和处理具体的信息，主要分析方式：E-R 模型(Entity-Relationship Model)图。

(3) 物理模型：针对逻辑模型中的内容，在具体的物理介质上实现出来(即在数据库中设计最终的数据表)。

其中上述三个步骤中逻辑模型和物理模型在开发中尤为重要，其中建立逻辑模型其实就是设计 E-R 图。E-R 图也称实体-联系图(Entity Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。E-R 图中包括四种符号：矩形、菱形、椭圆、连线，各种符号描述着不同的信息结构，具体如表 5.1 所示。

表 5.1 E-R 符号

符　　号	描　　述
	实体：客观存在的事物，一般是名词
	属性：用于描述实体的特性，每个实体可以有很多属性，一般是名词
	关系：反映两个实体之间客观存在的关系，一般是动词
	连接线：连接实体与属性以及实体与关系等

接下来以一个旅馆管理系统的 E-R 图来对表 5.1 进行较为详细的展示，如图 5.3 所示。

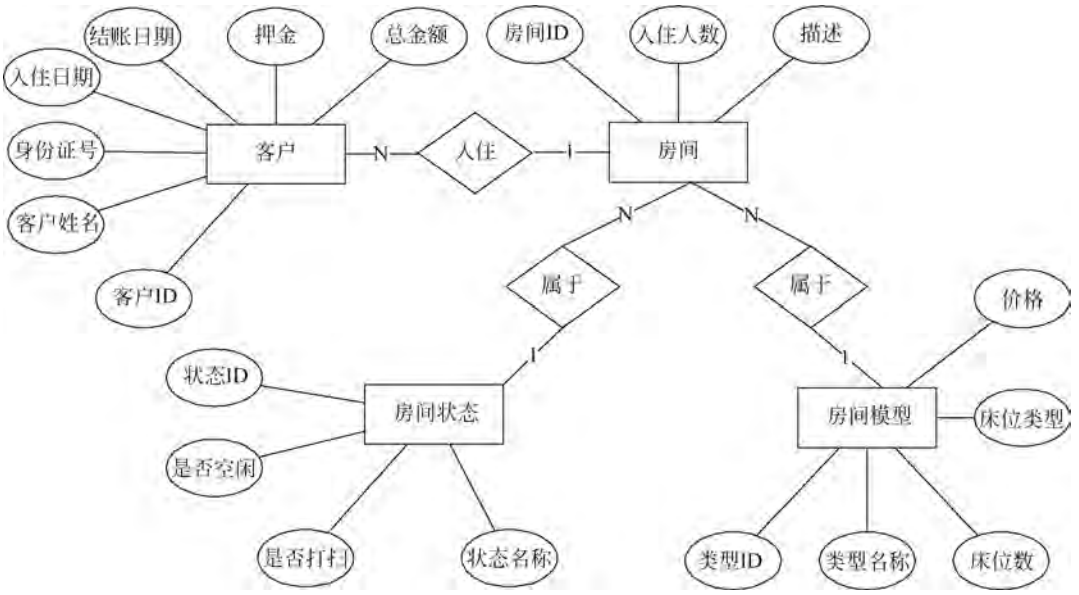

图 5.3　旅馆管理系统 E-R 图

在图 5.3 中，一共包含客户、房间、房间状态、房间类型 4 个实体，有联系的实体与实体之间的关系均是 1∶N。实体与实体之间存在三种对应关系，分别是一对一、一对多、多对多，具体如下：

- 一对一关系：实体 X 的任意一个实例至多有一个实体 Y 的实例与之对应，且实体 Y 的任意一个实例也至多有一个实体 X 的实例与之对应，例如，一个人只能有一张身份证。一对一关系在 E-R 图中被标记为 1∶1。
- 一对多关系：实体 X 的任意一个实例都可以对应任意数量实体 Y 的实例，而实体 Y 一个实例至多与实体 X 的一个实例对应，例如，一个班级有多名学生。一对多关系在 E-R 图中被标记为 1∶N。
- 多对多关系：实体 X 的任意一个实例都可以对应任意数量实体 Y 的实例，反之亦然，例如，学生和课程之间的关系，一个学生可以选多门课程，一门课程可以对应多名学生。多对多关系在 E-R 图中被标记为 N∶N。

在设计完 E-R 图之后，最终要将表设计编排到数据库中，这一步是物理模型的实现，即关系表设计。

将 E-R 图转换成关系表时需要以下步骤：

(1) 选定数据库，如 MySQL、Oracle、SQL Server 等，由于每种数据库所支持的数据类型可能略有差别，因此需要在设计表之前选定好数据库，本书使用 MySQL 进行设计。

(2) 将每一个实体转换成一个数据表，属性为数据表的列信息，并赋予对应的数据类型。

(3) 实体与实体之间关系为 1∶1 时，为两个表设置相同的主键。

(4) 实体与实体之间关系为 1∶N 时，在 N 对应的表中添加一个外键，并与 1 对应的表的主键相关联。

(5) 实体与实体之间的关系为 N∶N 时，需要生成一个单独描述关系的数据表，该关系

表的列包含两个实体表的主键。

(6) 最终审核所有表,添加相应的索引或约束。

按照上述步骤,图 5.3 中的旅馆管理系统可以转换为相应的数据表,上述图 5.3 中 E-R 图一共有 4 个实体,并且实体与实体之间关系均为 1∶N 关系,因此生成 4 个表,分别为 Customer 表、Room 表、Status 表、Style 表,具体如表 5.2～表 5.5 所示。

表 5.2 Customer 表

列 名	类 型	长 度	描 述	索引、约束
cid	Int	20	客户 id,主键,自动增长	PK Auto_increment
cname	Varchar	60	客户姓名	Not NULL
IDcard	Varchar	20	身份证号	Not NULL
indate	Datetime		入住时间	
outdate	Datetime		结账日期	
deposit	Float	20	押金	Not NULL
total_amount	Float	20	总金额	
rid	Int	20	房间 id,外键,对应 Room 表	FK

表 5.3 Room 表

列 名	类 型	长 度	描 述	索引、约束
rid	Int	20	房间 id,主键	PK
inperson	Int	10	入住人数	
description	Varchar	60	描述	
statusid	Int	20	状态 id,外键,对应 Status 表	FK
styleid	Int	20	类型 id,外键,对应 Style 表	FK

表 5.4 Status 表

列 名	类 型	长 度	描 述	索引、约束
statusid	Int	20	房间状态 id,主键	PK
statusname	Varchar	60	状态名称	
isempty	Int	4	是否空闲	
isclear	Int	4	是否打扫	

表 5.5 Style 表

列 名	类 型	长 度	描 述	索引、约束
styleid	Int	20	房间类型 id,主键,自动增长	PK Auto_increment
stylename	Varchar	60	类型名称	
bedstyle	Varchar	20	床的类型	
bedamount	Int	10	床位数	
price	Float	10	价格	

表 5.2(Customer 表)通过 rid(房间 id)这个外键来实现表 5.2 与表 5.3(Room 表)之间的联系(N∶1),表 5.3 通过 statusid(状态 id)和 styleid(类型 id)这两个外键实现了与表 5.4(Status 表)、表 5.5(Style 表)之间的联系(N∶1)。

通过上述数据表的转换，旅店管理系统的数据库设计部分基本完成，接下来将相关数据录入到相关数据表中。

5.2.3 SQL 简介

SQL(Structure Query Language，结构化查询语言)是专为数据库而建立的操作命令集，是一种功能齐全的数据库语言。在使用它时，只需要发出“做什么”的命令，“怎么做”是不用使用者考虑的。SQL 功能强大、简单易学、使用方便，已经成了数据库操作的基础，并且现在几乎所有的数据库都支持 SQL。

SQL 被美国国家标准局(ANSI)确定为关系型数据库语言的美国标准，后来被国际化标准组织(ISO)采纳为关系数据库语言的国际标准，各数据库厂商都支持 ISO 的 SQL 标准，并在标准的基础上做了不同的扩展。

从以上介绍可以看出，SQL 有以下几项优点：

(1) 不是某个特定数据库供应商专有的语言，几乎所有重要的数据库管理系统都支持 SQL。

(2) 简单易学，该语言的语句都是由描述性很强的英语单词组成，且这些单词的数目不多。

(3) 高度非过程化，即用 SQL 操作数据库，只需指出“做什么”，无须指明“怎么做”，存取路径的选择和操作的执行由数据库自动完成。

SQL 包含了所有对数据库的操作，它主要由四个部分组成，具体如下：

(1) 数据库定义语言(DDL)：用于定义数据库、数据表等，其中包括 CREATE 语句、ALTER 语句和 DROP 语句，CREATE 语句用于创建数据库、数据表等，ALTER 语句用于修改表的定义等，DROP 语句用于删除数据库、删除表等。

(2) 数据库操作语言(DML)：用于对数据库进行添加、修改和删除操作，其中包括 INSERT 语句、UPDATE 语句和 DELETE 语句，INSERT 语句用于插入数据，UPDATE 语句用于修改数据，DELETE 语句用于删除数据。

(3) 数据库查询语言(DQL)：用于查询，也就是 SELECT 语句，SELECT 语句可以查询数据库中的一条或多条数据。

(4) 数据控制语言(DCL)：用于控制用户的访问权限，包括 GRANT 语句、REVOKE 语句、COMMIT 语句和 ROLLBACK 语句，GRANT 语句用于给用户添加权限，REVOKE 语句用于收回用户的权限，COMMIT 语句用于提交事务，ROLLBACK 语句用于回滚事务。

通过 SQL 可以直接操作数据库，很多应用程序中也经常使用 SQL 语句，例如 Python 程序中可以嵌入 SQL 语句，实现 Python 程序调用 SQL 语句操作数据库，此外，PHP、Java 等语言也可以嵌套 SQL。

5.2.4 SQL 实战

上节主要内容是讲解 SQL，本节带领大家学习 SQL 语句的不同功能实现。

1. Create 语句

Create 语句可以创建数据库和数据表，其中创建数据库的语法格式如下：

```
CREATE DATABASE database_name;
```


上述示例中，database_name 是数据库名称，其中旅馆管理系统所创建的数据库名称为“hotel”。

create table 语句是新建数据表语句，主要实现创建新的数据表，其语法格式如下：

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

上述示例中，table_name 是表名，column_name1 是列名，data_type 是数据类型，size 是数据的长度。接下来以上述旅馆管理系统中 Status 表和 Style 表的创建为示例讲解 create table 语句，具体示例如下：

```
/* 创建 Status 表 */
CREATE TABLE Status
(
 statusid int(11) ,
 statusname VARCHAR(60),
 isempty INT(11),
 isclear INT(11),
 PRIMARY KEY (statusid)
);
/* 创建 Style 表 */
CREATE TABLE Style
(
 styleid int(11) AUTO_INCREMENT,
 stylename VARCHAR(60),
 bedstyle VARCHAR(20),
 bedamount INT(10),
 price      FLOAT(10),
 PRIMARY KEY (styleid)
);
```

上述示例中，实现了创建表 Status 和表 Style，并对表中各字段进行设置约束。

2. Insert 语句

Insert 语句是插入语句，作用是向数据表中插入相关数据，其语法结构如下：

```
INSERT INTO table_name(列名 1,列名 2,列名 3,...)
VALUES(值 1,值 2,值 3,...)
```

上述结构是向数据表中按列按值插入数据，当没指明列名时，默认是按表中列的排列顺序进行数值插入，具体示例如下：

```
/* 默认插入方式 */
INSERT INTO Room
VALUES(1,2,'豪华间,一张双人床',1,1)
/* 按标准方式插入 */
INSERT INTO Room(rid,inperson,decription,statusid,styleid)
VALUES(2,2,'标准间,两张单人床',1,2)
/* 按指定列顺序插入 */
INSERT INTO Room(rid,statusid,styleid,inperson,decription)
VALUES(3,1,3,1,'单人间,一张单人床')
```

上述示例的三种方式都能实现向 Room 表中插入相关数据,第一种是默认方式插入;第二种是标准方式插入;第三种是按指定顺序插入;在执行 Insert 操作时,当对应列的数据未指定时,会将该字段的数据设置成默认值。

注意:SQL 中关键字不区分大小写,如 INSERT 与 insert 作用相同。

3. Delete 语句

Delete 语句是删除语句,作用是将数据表中的数据删除,其语法结构如下:

```
DELETE FROM table_name
WHERE 条件
```

上述代码是将 table_name 表中满足 where 条件的数据删除,具体示例如下:

```
/* 删除类型 id 为 8 的房间类型 */
DELETE FROM Style
WHERE styleid = 8
/* 删除价格小于 200 或大于 10 000 的房间类型 */
DELETE FROM Style
WHERE price > 10 000 OR price < 200
/* 删除类型名字以单人开头的房间类型 */
DELETE FROM Style
WHERE stylename LIKE '单人%'
```

上述示例列举了三个示例,第一个是将 Style 表中类型 id 为 8 的房间类型删除,第二个是将 Style 表中价格小于 200 或大于 10 000 的房间类型删除,第三个是将 Style 表中以“单人”开头的房间类型删除。

4. Update 语句

Update 语句是更新语句,作用是更新数据表中数据,其语法结构如下:

```
UPDATE table_name
SET 列名 1 = 新值,列名 2 = 新值,列名 3 = 新值...
WHERE 条件
```

上述代码是通过 Update 语句将 table_name 表中将满足 where 条件的列名 1、列名 2、列名 3 等的值修改为新值,具体示例如下:

```
UPDATE Status
SET isempty = 2, isclear = 2
WHERE statusid = 2
```

上述示例是将 Status 表中 statusid 等于 2 的 isempty 和 isclear 字段的值更新为 2。

5. Select 语句

select 语句是查询语句，主要实现从数据库中提取相关数据，语法结构如下：

```
SELECT [distinct|top] 列名 1,列名 2... FROM table_name
[WHERE 条件]
[GROUP BY 列名 [HAVING 分组筛选条件] ]
[ORDER BY 列名 1 [ASC|DESC],列名 2[ASC|DESC]...]
```

上述结构“[ ]”中的内容为可选项，WHERE 语句是条件控制，GROUP BY 语句是以某列进行分组，HAVING 对分组进行条件筛选，ORDER BY 语句则是对查询之后的数据按照某一列或多列进行排序，具体示例如下：

```
/* 查询 Customer 表中所有记录 */
SELECT * FROM Customer
/* 查询 Customer 表中所有客户姓名,并排除相同名字 */
SELECT DISTINCT cname FROM Customer
/* 查询 Customer 表中消费总金额大于 10 000 的前 5 条记录 */
SELECT top 5 * FROM Customer WHERE total_amount > 10000
/* 查询 Customer 表中以 rid 分组并返回超过 10 个客户的客户人数 */
SELECT COUNT(cname) FROM Customer
GROUP BY rid HAVING COUNT(cname)> 10
```

上述示例使用了 SQL 中的聚集函数 COUNT()，其他常用的 SQL 聚集函数如表 5.6 所示。

表 5.6　SQL 中常用的聚集函数

聚 集 函 数	描　　述
AVG(col)	返回某列的平均值
COUNT(col)	返回某列的行数
MAX(col)	返回某列的最大值
MIN(col)	返回某列的最小值
SUM(col)	返回某列值之和
FIRST(col)	返回某列的第一个值
LAST(col)	返回某列的最后一个值

在 where 语句和 having 语句中还使用了 SQL 中常用的条件操作符“>”，其他常用的 SQL 条件操作符如表 5.7 所示。

表 5.7　SQL 中常用的条件操作符

操作符	描　　述	举　　例
>	大于	'total_amount'> 10000
<	小于	'price'< 1000
=	等于	'cname'= '千锋'
<>	不等于	'cname'<>'1000phone'
>=	大于等于	'price'>= 500
<=	小于等于	'price'<= 500
Between	在两个数之间	'price'Between 500 and 1000
In	是否在集合中	'cname'in('千锋''好程序员''1000phone''扣丁学堂')
Like	模糊匹配	'cname'like'千锋%'//匹配以'千锋'为开头客户姓名
IS NULL	判断是否为空	'IDcard'IS NULL
AND	并,用于连接多个条件表达式	'price'<= 1000 AND'deposit'<= 500
OR	或,用于连接多个条件表达式	'price'= 1000 OR 'price'= 500

注意：当 AND 与 OR 同时出现时,AND 的优先级高于 OR,但是可以通过小括号"()"来改变优先级。

上述 select 语句都是单表查询,在实际开发过程中会涉及多表查询,实现多表查询的方式有很多种,本节主要讲解关键字 join 在多表查询中的使用,其语法格式如下：

```
SELECT 列名 1,列名 2,...
FROM table_name1 JOIN table_name2 ON 连接表达式
WHERE ...(条件表达式)
```

上述语法结构是通过 join…on 关键字将 table_name1 与 table_name2 连接起来,并从连接之后的表中按 where 条件查询出列名 1,列名 2…等内容。join 关键字还有很多种类型,如表 5.8 所示。

表 5.8　JOIN 关键字的类型表

类　　型	含　　义
INNER JOIN	内连接,获取两个表中满足条件的不重复的数据记录
LEFT [OUTER] JOIN	左外连接,返回左表的所有行,不仅仅是连接列所匹配的行,右表中无匹配的数据记录则显示空值
RIGHT [OUTER] JOIN	右外连接,返回右表的所有行,不仅仅是连接列所匹配的行,左表中无匹配的数据记录则显示空值
FULL[OUTER] JOIN	完整外部连接,返回左表和右表中的所有行,即左外连接与右外连接以及内连接的结果合集

具体示例如下：

```
/* 查询押金不为零的客户信息以及所入住房间信息 */
SELECT c.*,r.rid,r.inperson
FROM Customer c LEFT JOIN Room r ON c.rid = r.rid
WHERE c.deposit <> 0
```

上述示例是通过左外连接实现查询押金不为 0 的客户信息以及入住房间信息。

6. Python 中 SQL 的实战使用

SQL 标准统一了数据库语言，对于不同数据库都可以实现操作，但在一些高级语言（如：Python、Java、C++等）当中使用时，还需要连接每个数据库的引擎，才能使用 SQL 语言对数据库进行操作。在 Python 中通过引入不同的数据包来实现不同的数据库，常用数据库的 Python 包如表 5.9 所示。

表 5.9　常用数据库引擎的 Python 包

数　据　库	对应 Python 包
MySQL	MySQLdb
SQLite	SQLite3
Oracle	cx_Oracle
PostgreSQL	PsyCopg2 或 PyPgSQL 或 PyGreSQL
MS SQL Server	pymssql
Excel	pyExcelerator

接下来以 MySQL 数据库为例演示 SQL 语言在 Python 中应用，在讲解实例之前，需要先在 PyCharm 中安装 MySQL 对应的 Python 包——PyMySQL，具体步骤如下：

（1）单击 File 并找到 Settings，如图 5.4 所示。

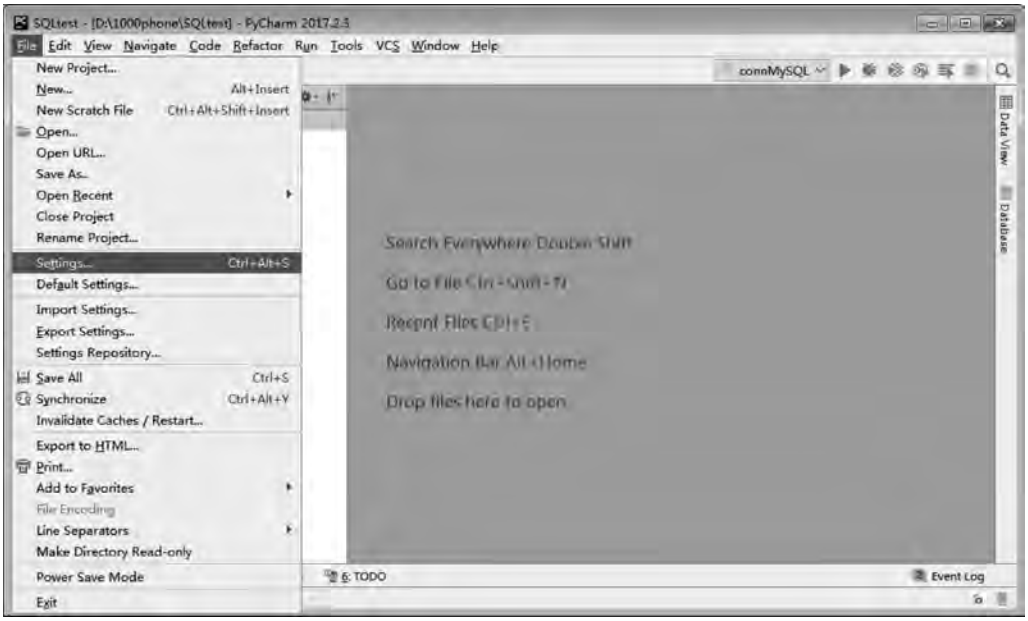

图 5.4　【File】界面

（2）单击 Settings，进入如图 5.5 所示界面。

（3）找到【Project：项目名】，并单击 Project Interpreter，然后再单击右上角绿色的【＋】号进入如图 5.6 所示界面。

（4）在搜索栏中输入 PyMySQL，并找到 PyMySQL，并单击左下角 Install Package，安

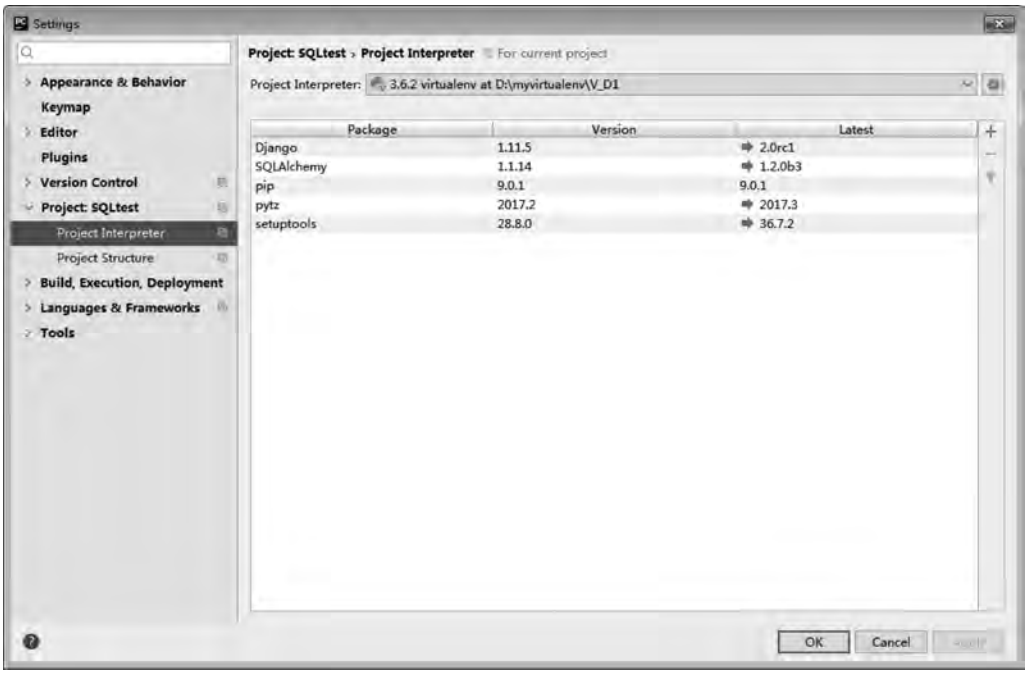

图 5.5　Settings 界面

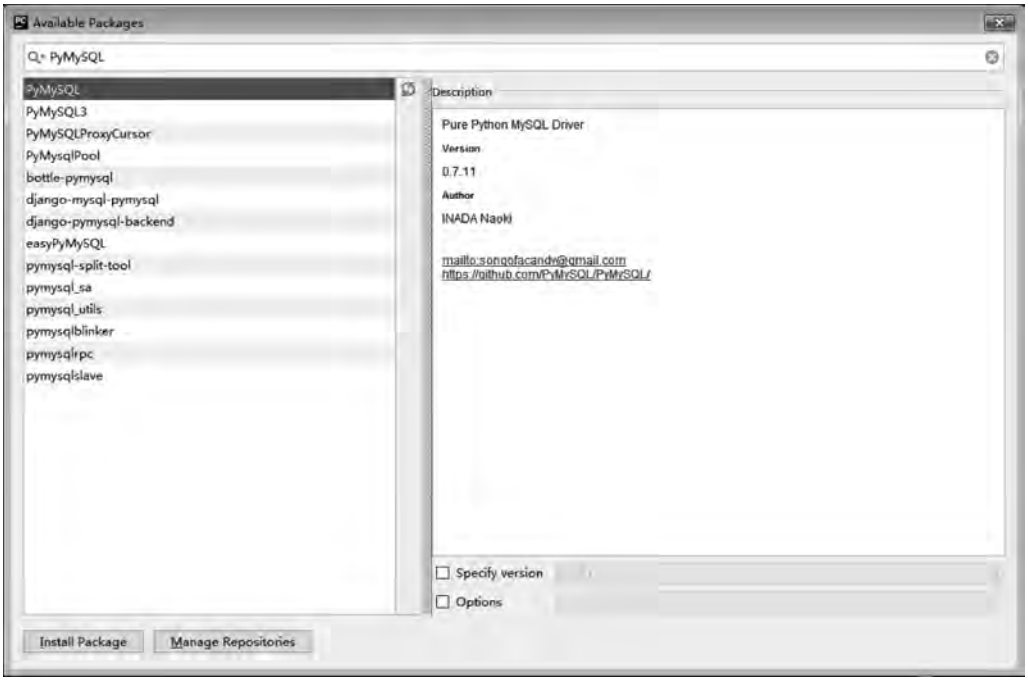

图 5.6　搜索【PyMySQL】包界面

装成功后如图 5.7 所示。

（5）返回到前一个界面就可以看到 PyMySQL 已经安装成功，如图 5.8 所示。

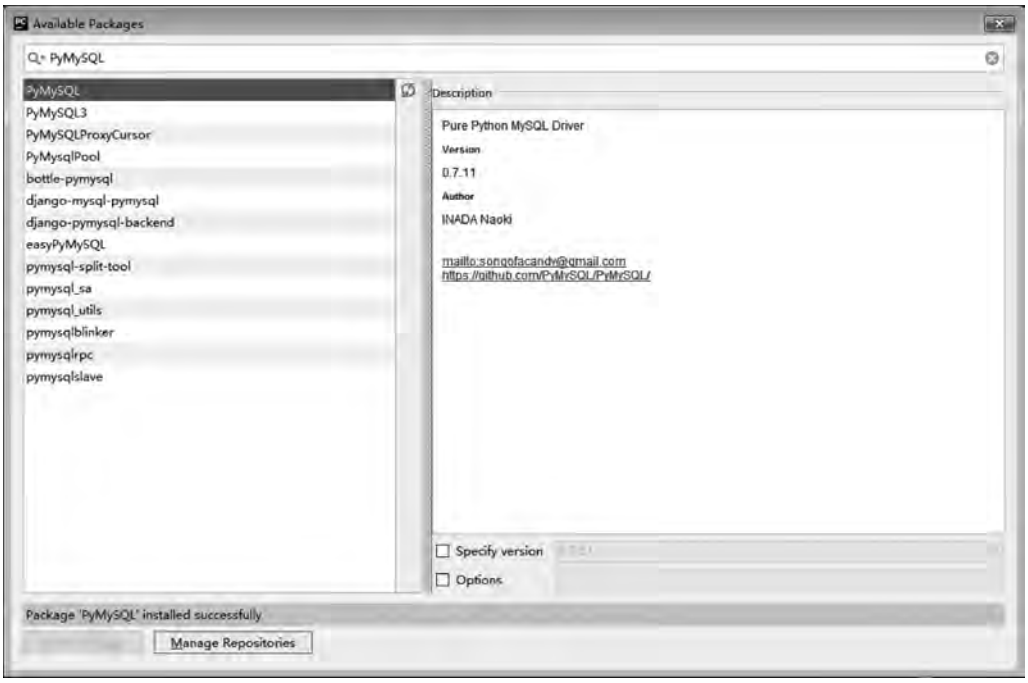

图 5.7 【PyMySQL】安装成功界面

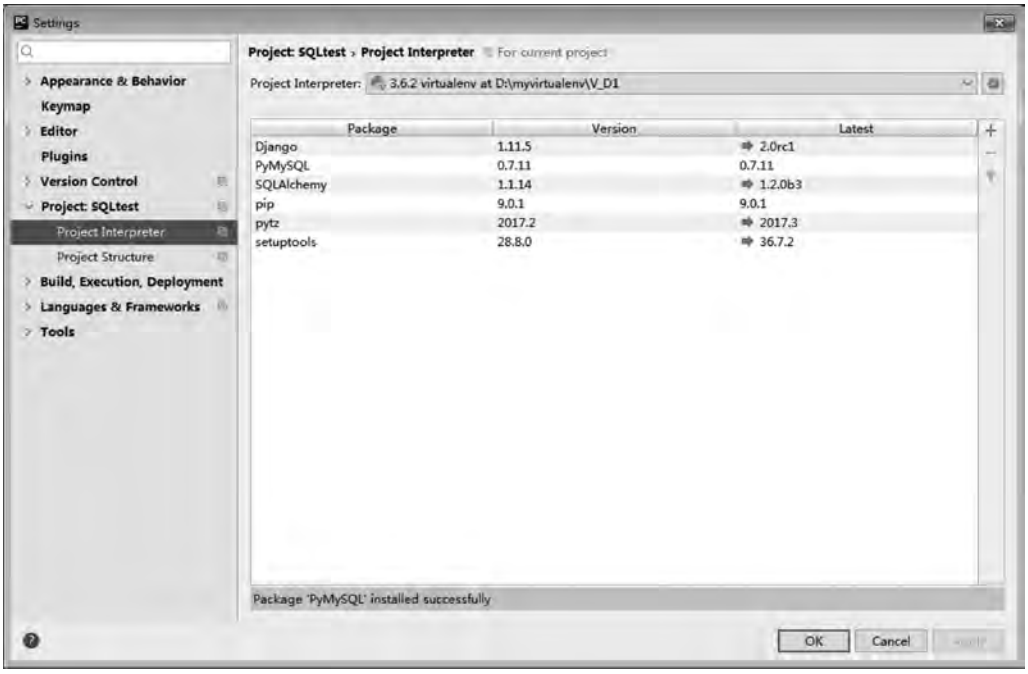

图 5.8 安装成功界面

安装好 PyMySQL 之后开始进行案例的实现，如例 5-1～例 5-4 所示。

【例 5-1】 MySQL 数据库为例演示 SQL 语言在 Python 中应用(1)

```
import pymysql
#建立数据库连接
conn = pymysql.Connect(
    host = 'localhost',
    port = 3306,
    user = '1000phone',
    passwd = '1000phone',
    db = 'hotel',
    charset = 'utf8'
)
#获取游标
cursor = conn.cursor()
#数据库中插入数据
sql_insert = "INSERT INTO Style(stylename,bedstyle,bedamount,price) " \
             "values('单人间','单人高级套房','1','1000')," \
             "('单人间','单人高级套房','1','1000')," \
             "('普通单人间','普通单人床','1','200')," \
             "('普通双人间','普通双人床','1','1000')," \
             "('豪华双人间','豪华双人床','1','2000')," \
             "('一般双人间','普通单人床','2','800')," \
             "('总统套房双人间','总统套餐双人床','1','5000')"
#执行语句
cursor.execute(sql_insert)
#事务提交,否则数据库得不到更新
conn.commit()
#数据库连接和游标的关闭
conn.close()
cursor.close()
```

运行之后通过 Navicat(MySQL 数据库的一个第三方可视化软件,有需要请自行安装)查看 Style 表,如图 5.9 所示。

styleid	stylename	bedstyle	bedamount	price
1	单人间	单人高级套房	1	1000
2	单人间	单人高级套房	1	1000
3	普通单人间	普通单人床	1	200
4	普通双人间	普通双人床	1	1000
5	豪华双人间	豪华双人床	1	2000
6	一般双人间	普通单人床	2	800
7	总统套房双人间	总统套餐双人床	1	5000

图 5.9　运行之后的 Style 表

【例 5-2】 MySQL 数据库为例演示 SQL 语言在 Python 中应用(2)

```
import pymysql
#建立数据库连接
conn = pymysql.Connect(
    host = 'localhost',
    port = 3306,
    user = '1000phone',
```

```
    passwd = '1000phone',
    db = 'hotel',
    charset = 'utf8'
)
#获取游标
cursor = conn.cursor()
#修改数据库中的内容
sql_update = "UPDATE Style SET stylename = '豪华单人间'," \
             "bedstyle = '豪华单人床',price = 500 " \
             "WHERE styleid = 1"
cursor.execute(sql_update)
conn.commit()
#数据库连接和游标的关闭
conn.close()
cursor.close()
```

运行之后通过 Navicat 查看 Style 表，如图 5.10 所示。

styleid	stylename	bedstyle	bedamount	price
1	豪华单人间	豪华单人床	1	500
2	单人间	单人高级套房	1	1000
3	普通单人间	普通单人床	1	200
4	普通双人间	普通双人床	1	1000
5	豪华双人间	豪华双人床	1	2000
6	一般双人间	普通单人床	2	800
7	总统套房双人间	总统套餐双人床	1	5000

图 5.10　运行之后的 Style 表

【例 5-3】 MySQL 数据库为例演示 SQL 语言在 Python 中应用(3)

```
import pymysql
#建立数据库连接
conn = pymysql.Connect(
    host = 'localhost',
    port = 3306,
    user = '1000phone',
    passwd = '1000phone',
    db = 'hotel',
    charset = 'utf8'
)
#获取游标
cursor = conn.cursor()
#删除数据库中的内容,并利用 try catch 语句进行事务回滚
try:
```

```
    sql_delete = "DELETE FROM Style WHERE styleid = 2"
    cursor.execute(sql_delete)
    conn.commit()
except Exception as e:
    print(e)
    #事务回滚,即出现错误后,不会继续执行,
    #而是回到程序未执行的状态,原先执行的也不算了
    conn.rollback()
#数据库连接和游标的关闭
conn.close()
cursor.close()
```

运行之后通过 Navicat 查看 Style 表,如图 5.11 所示。

styleid	stylename	bedstyle	bedamount	price
1	豪华单人间	豪华单人床	1	500
3	普通单人间	普通单人床	1	200
4	普通双人间	普通双人床	1	1000
5	豪华双人间	豪华双人床	1	2000
6	一般双人间	普通单人床	2	800
7	总统套房双人间	总统套餐双人床	1	5000

图 5.11　运行之后的 Style 表

【例 5-4】 MySQL 数据库为例演示 SQL 语言在 Python 中应用(4)

```
import pymysql
#建立数据库连接
conn = pymysql.Connect(
    host = 'localhost',
    port = 3306,
    user = '1000phone',
    passwd = '1000phone',
    db = 'hotel',
    charset = 'utf8'
)
#获取游标
cursor = conn.cursor()
#从数据库中查询
sql = "SELECT * FROM Style"
#cursor 执行 sql 语句
cursor.execute(sql)
#打印执行结果的条数
print(cursor.rowcount)
#使用 fetch 方法进行遍历结果
rr = cursor.fetchall()  # 将所有的结果放入 rr 中
#对结果进行处理
for row in rr:
   print("房间 id 是: %d, 类型名称是: %s,"
         " 床的类型是: %s,床的数量是: %d, "
         "价格是: %.2f" % row)
```

```
26  #数据库连接和游标的关闭
27  conn.close()
28  cursor.close()
```


例 5-4 运行结果如图 5.12 所示。

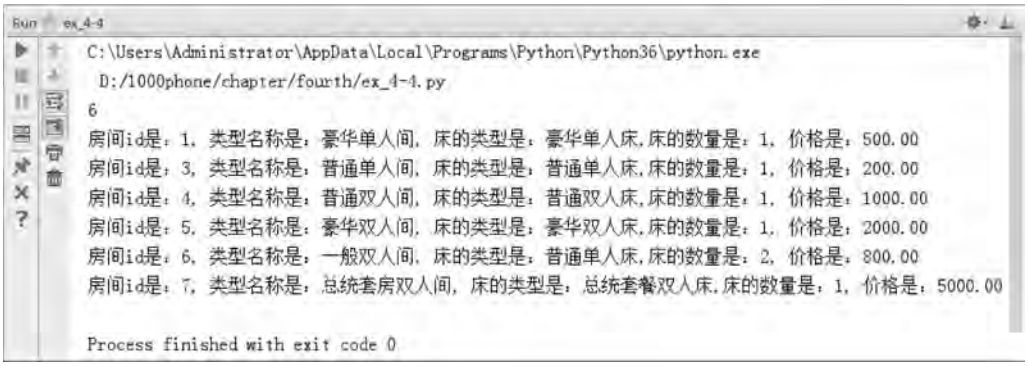

图 5.12　例 5-4 运行结果

5.3　Redis 安装

上述内容的操作都是在 MySQL(关系型数据库)下进行，在实际开发过程中还会常用到非关系型数据库，如 Redis，因此本节主要讲解 Redis 数据库的安装，对 Redis 数据的操作在后续章节有相应介绍。

Redis 的具体安装步骤如下：

(1) 下载 Redis，地址为 https://github.com/MSOpenTech/redis/releases，下载 Redis-x64-3.2.100.zip，如图 5.13 所示。

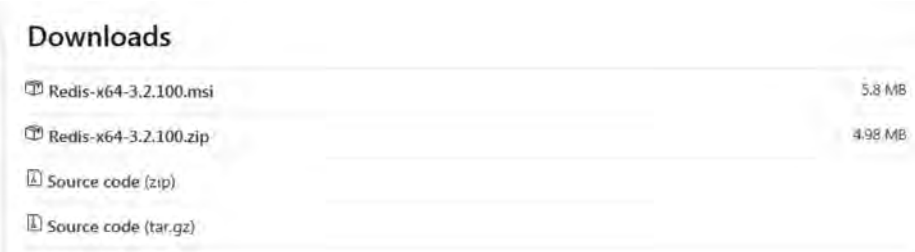

图 5.13　Redis 下载

(2) 直接解压到存储目录，本书存储在 D:\software\redis 中。

(3) 打开命令提示符(Win 键＋R，输入 cmd)，进入 Redis 的存储目录。

(4) 输入 redis-server.exe redis.windows.conf 命令启动 Redis 服务器端，如图 5.14 所示。

在图 5.14 中，Redis 成功启动，说明 Redis 数据库安装成功。

图 5.14　启动 Redis

5.4　本章小结

本章主要对 Web 开发中的核心内容——数据库进行讲解，其中包括数据库基本概念、关系型数据库、Redis 安装三部分内容，本章重点讲解了数据库的基本设计、SQL 基础与实战以及 Redis 数据库的安装。数据库是 Web 开发中数据的提供方，没有数据，Web 网站就没有实际意义，因此大家一定要将数据库的内容反复学习，最终熟练运用。

5.5　习　　题

1. 填空题

(1) 数据库系统包含_____、_____、_____三部分内容。

(2) E-R 图中“实体”指的是_____。

(3) 实体之间存在三种联系，分别是_____、_____、_____。

(4) SQL 语句中进行查询的关键字是_____。

(5) SQL 语句中关键字_____大小写。

2. 选择题

(1) SQL 中，(　　)操作符是用来进行模糊匹配。

A. In　　B. Between　　C. OR　　D. Like

(2) 下列可以用来对查询结果排序的是(　　)。

A. order by　　B. Having　　C. group by　　D. where

(3) 在 E-R 图中,(　　)符号表示实体。

A. 连线　　B. 矩形　　C. 菱形　　D. 椭圆

(4) 下列聚集函数中返回某列平均数的函数是(　　)。

A. COUNT(col)　　B. MAX(col)　　C. SUM(col)　　D. AVG(col)

(5) 以下不属于关系型数据库的是(　　)。

A. Oracle　　B. MongoDB　　C. MySQL　　D. DB2

3. 思考题

(1) 简述数据库的主要特点。

(2) 常见的数据库有哪些?

4. 编程题

创建一个学生表,并实现学生信息的增删改查。